What is claimed is:

1. A page fault proxy handler for connection to an original page fault handler and a paging table in which supervisor flags for all entries for all writable memory pages have been pre-set, the page fault proxy handler comprising:

a page fault detector;

a mitigation module;

a page fault filter, connected to the page fault detector, wherein the filter passes to the original page fault handler page faults not arising from an attempt to access a writable page by a user mode program;

a controlled memory access module, wherein the controlled memory access module permits a user program to access a writable page of memory by changing an associated supervisor flag in the paging table; and

an execution address checker, connected to the page fault filter, the mitigation module and the controlled memory access module, wherein the execution address checker passes to the mitigation module only page faults arising from an attempt by a user mode program to execute from a predetermined section of executable memory, and wherein the execution address checker passes to the controlled memory access module page faults arising from any other attempt by a user mode program to access a writable page.

2. The page fault proxy handler of claim 1 wherein the paging cache is a data translation lookaside buffer.

3. The page fault proxy handler of claim 1 wherein the predetermined section of executable memory is a stack.

4. The page fault proxy handler of claim 1 wherein the predetermined section of executable memory is all executable memory.

5. The page fault proxy handler of claim 1 wherein the mitigation module comprises a code termination module.

6. The page fault proxy handler of claim 1 wherein the mitigation module comprises a logging module.

7. The page fault proxy handler of claim 1 wherein the apparatus is for use with an IA-32 microprocessor.

8. A method for handling page faults, for use with an original page fault handler, the method comprising:

setting a supervisor flag in a page entry table associated with a writable page;

detecting a page fault;

determining whether the page fault arises from an attempt by a user mode program to access the writable page having the associated supervisor flag set; and

conditionally calling the original page fault handler on the basis of the determining step.

9. The method of claim 8 further comprising:

providing a page fault proxy handler that performs the detecting determining and conditionally calling steps.

10. The method of claim 9 further comprising:

launching the page fault proxy handler with one or more runtime options.

11. The method of claim 10 wherein the runtime options affect the performance overhead and/or security efficacy of the page fault proxy handler.

12. The method of claim 8, further comprising:

determining whether the page fault was caused by an attempt to execute from the page.

13. The method of claim 12 wherein the page fault is associated with a fault address, and wherein the step of determining whether the page fault was caused by an attempt to execute from the page comprises comparing the fault address to the contents of an instruction pointer.

14. The method of claim 12 further comprising:

if the page fault was not caused by an attempt to execute from the page, then performing at least the following steps:

clearing the supervisor flag in a paging cache associated with the page;

accessing the page after the clearing step; and

setting the supervisor flag after the accessing step.

15. The method of claim 14 wherein the paging cache is a data translation lookaside buffer.

16. The method of claim 12 further comprising:

terminating the user mode program, if the page fault was caused by an attempt to execute from the page.

17. The method of claim 16 wherein the terminating step comprises:

injecting termination code in the user mode program; and

changing a return address.

18. The method of claim 16 wherein the terminating step comprises:

prompting an operator whether to terminate the user mode program; and

accepting a response from the operator.

19. The method of claim 16 wherein the terminating step comprises:

logging an event, if a fault address equals a current execution address.

20. The method of claim 8 further comprising:

determining whether the page fault arises in a predetermined section of memory.

21. The method of claim 20 wherein the predetermined section of memory is all memory.

22. The method of claim 20 wherein the predetermined section of memory is a stack.

23. The method of claim 8 further comprising:

checking whether the page fault is for an existing page of memory.

24. The method of claim 8 further comprising:

checking whether the page fault is for a kernel page of memory.

25. The method of claim 8 wherein the method is performed with an IA-32 microprocessor.

26. An apparatus for use with an original page fault handler, the apparatus comprising:

a means for setting a supervisor flag in a page table associated with a writable page;

a means for detecting a page fault;

a means for determining whether the page fault arises from an attempt by a user mode program to access the writable page having the associated supervisor flag set; and

a means for conditionally calling the original page fault handler on the basis of the determining step.

27. A computer readable medium on which is embedded computer software, the software performing a method for handling page faults, for use with an original page fault handler, the method comprising:

setting a supervisor flag in a page entry table associated with a writable page;

detecting a page fault;

determining whether the page fault arises from an attempt by a user mode program to access the writable page having the associated supervisor flag set; and

conditionally calling the original page fault handler on the basis of the determining step.